Multi-Learner based Recursive Supervised Training

[1]Laxmi R. Iyer, [1]Kiruthika Ramanathan, and [2]Sheng-Uei Guan

[1]Department of Electrical and Computer Engineering,
National University of Singapore, 4 Engineering Drive 3, Singapore 117576
[2] School of Engineering and Design
Brunel University, Uxbridge, Middlesex, UB8 3PH, UK

Abstract

In this paper, we propose the Multi-Learner Based Recursive Supervised Training (MLRT) algorithm which uses the existing framework of recursive task decomposition, by training the entire dataset, picking out the best learnt patterns, and then repeating the process with the remaining patterns. Instead of having a single learner to classify all datasets during each recursion, an appropriate learner is chosen from a set of three learners, based on the subset of data being trained, thereby avoiding the time overhead associated with the genetic algorithm learner utilized in previous approaches. In this way MLRT seeks to identify the inherent characteristics of the dataset, and utilize it to train the data accurately and efficiently. We observed that empirically, MLRT performs considerably well as compared to RPHP and other systems on benchmark data with 11% improvement in accuracy on the SPAM dataset and comparable performances on the VOWEL and the TWO-SPIRAL problems. In addition, for most datasets, the time taken by MLRT is considerably lower than the other systems with comparable accuracy. Two heuristic versions, MLRT-2 and MLRT-3 are also introduced to improve the efficiency in the system, and to make it more scalable for future updates. The performance in these versions is similar to the original MLRT system.

Keywords: Neural Networks, Supervised Learning, Probabilistic Neural Networks (PNN), Backpropagation

1. Introduction

Consider a scenario where a child learns to classify animals. Under supervision with labeled examples, he uses his learners, i.e. sight, audio, smell, touch, etc. one by one to explore the examples. He decides that sight can enable him to pick out the most dominant characteristics, and utilizes that in learning the examples which are best learnt using the sight learner. He then focuses on examples which he has not learnt in the first round. He again explores with all his learners to pick out characteristics that are now prominent, and picks audio as the learner for this round of learning. This process continues, till he is able to identify unseen animals.

Similarly, this process is used to understand real world objects and is so intuitive that one is hardly aware of it. This paper proposes a method that uses a similar approach for supervised learning, based on neural networks.

In neural networks there are many single learner systems, i.e. systems that use one method to classify all problems. Although these systems do well on some problem sets, they perform poorly when dealing with others. For example, simulations on the TWO-SPIRAL dataset show that using the Probabilistic Neural Network (PNN) has an error of less than 13%, which is comparable to most benchmark algorithms. However, the PNN network obtains a 28% error with the SPAM dataset, while other popular algorithms can perform much better[1]. Instead of trying to learn all datasets with just one method, the Multi-Learner Based Recursive Supervised Training (MLRT) has three learners, Bounding Boxes (BB), Backpropagation [1-2], and Probabilistic Neural Networks (PNN)

[1] The simulation results are presented in further detail at the end of this paper.

[3-5], each of which works best for different type of problems. A learner most suited for the problem set is first chosen and the subset of data that can be classified accurately with this learner is picked out. The subset is trained by the learner. This process is then repeated with the rest of the data, till the number of remaining patterns is too few.

It is not possible to cater specifically to every single dataset, as there are an infinite variety of problems in the real world. However, the aim is to be able to cater to as many datasets as possible. The most simple classification problems are those where different classes can be separated by a boundary. Backpropagation is able to solve classification problems where the classes are separable. In more complex problems, there are several separable regions of the same class. If these regions are labeled differently, then they would be separable before passing it through a BP. This different labeling is done before classification (using BP) in the BB subsystem. Other times, separable classes are represented by different curves, which are tackled by the PNN curve-fitting network. Most real life data would be more complex, but in this system, we use these simple systems, i.e. BP, BB and PNN as barebones to classify more complex problems in a recursive manner. In addition to solving the problem, these subsystems give useful information on the data, which can be used for future updates.

The paper is organized as follows. In Section 2 we discuss methods related to this work. Section 3 describes the MLRT architecture in detail. In Section 4, we analyze the complexity theoretically and prove the convergence of MLRT. Section 5 provides simulation results. Section 6 explores the performance of the current MLRT system with

one where BB is replaced by clustering [6]. Section 7 introduces MLRT-2 and MLRT-3. In Section 8, we provide some discussions, followed by Section 9, the conclusion.

2. Related Work

MLRT uses three learners, i.e. the Bounding Boxes (BB) which is an alternative to clustering [6-13], Backpropagation, and Probabilistic Neural Networks. In addition, it is based on the framework of RPHP [14]. MLRT performs task decomposition the way RPHP does, i.e. by choosing the best learnt patterns, training them, and repeating the method with the rest of the dataset. These related systems would be explained in this section.

2.1 Backpropagation

Using a forward propagation of outputs layer by layer, and a backward propagation of errors, backpropagation adjusts the weights in order to reduce the error. This continues until the error is at local minima (gradient descent method). Backpropagation has universal approximation capability for classification problems, where the classes are separable. However, for complicated problems it requires an arbitrary choice of neuron number and architecture, and the absence of the correct value may cause inaccuracy.

2.2 Probabilistic Neural Networks

Probablistic Neural Networks are derived from Radial Basis Functions [15-16], which are typically used for regression problems. PNNs can be used to represent both an interpolation or approximation curve. They need to be manually tuned in the case of

noisy data, and may not provide good generalization accuracy where the curve is not characteristic of the dataset.

2.3 Clustering

Clustering algorithms are used in unsupervised learning to identify the natural clusters created by the dataset. While conventional means of clustering are able to identify clusters based on several criteria, including minimal distance (Kmeans [10-11], SOMs [12-13]), and parametric criteria (Agglomerative Hierarchical Clustering (AHC) [9]), the clusters identified may not be linearly separable. The Bounding Boxes proposed in this paper overcome this shortcoming and guarantee linear separability.

The above methods, although effective, find one good solution to the entire dataset. It is possible on the other hand, to divide the data into smaller subsets, find solutions to these subsets, and combine them into one. Recent algorithms attempt to do this by dividing the data according to class labels [17, 22]. However, there is no assurance that a set of two-class problems would be simpler than a multi class problem. RPHP overcomes this problem by choosing the subsets of data which are best trained for the network.

2.4 RPHP

The Recursive Percentage Based Hybrid Pattern Training (RPHP) [14] uses genetic algorithms (GA) [18-19] for training the datasets. The best trained patterns are chosen, and a network is created. This process repeats with the rest of the patterns. Although RPHP can attain high accuracy, it uses a GA to perform data decomposition. GA is a

blind search approach and is therefore limited in its flexibility and modification capability. We therefore aim, with the proposed MLRT approach, to overcome this limitation of GA and to further improve the flexibility of RPHP.

2.5. Literature review

MLRT is essentially a task decomposition technique where the dataset is decomposed recursively into subsets, and each subset is trained separately by a learner. In addition to RPHP, several methods have been proposed in literature, which deal with supervised learning by using task and data decomposition methods to simplify the problem.

Ensemble Learning: An ensemble of learners is a set of learners whose individual decisions are combined in some way (using either weighted or unweighted voting) to classify new samples. Ensemble learning is based on the assumption that "several minds are better than one".

The basic ensemble is created using Bayesian averaging [25]. However, more recent ensembles have been shown to be highly effective. Boosting [24] and bagging [26] introduce diversity in the learners by manipulating the training samples.

Class Based Task decomposition: Output parallelism [22] was proposed to reduce the training complexity by dividing the training data into subsets according to the output classes. A subnetwork is then trained using each subset of data, thereby simplifying the training data complexity. The system therefore consists of a series of subneural networks which are combined together to solve the problem.

Limitations: While all the above algorithms are effective algorithms, each of them has strengths and drawbacks. The accuracy of boosting and bagging is shown to depend on the number of weak learners used, this number being problem dependent [24]. Output

parallelism and related classwise decomposition algorithms [22] pre-partition the dataset according to class labels. The assumption is that a two-class problem is generally easier to solve than a K-class problem, However it can be applied to classification problems only and therefore limited in nature.

3. System Architecture

The following terms would be useful in explaining the system design. They will become clearer, as the system is explained.

- *Accuracy Detection* – 1st stage of training where the dataset is sent to the three subsystems, each of which returns the percentage accuracy and indices. (These terms are explained below.)
- *Network Creation* – 2nd stage of training where the network is created.
- *Subsystem* – refers to the learners, i.e. Backpropagation, Bounding Boxes, and Probabilistic Neural Networks (PNN).
- *Bounding Boxes (BB)* – One of the subsystems. It divides the data into hyper-dimensional boxes, and would be explained in detail at the end of this section.
- *Percentage accuracy (PA)* – percentage of data that can be accurately classified by each subsystem.
- *Indices* – the indices of the chosen patterns in the training data array. These are patterns which are accurately classified.

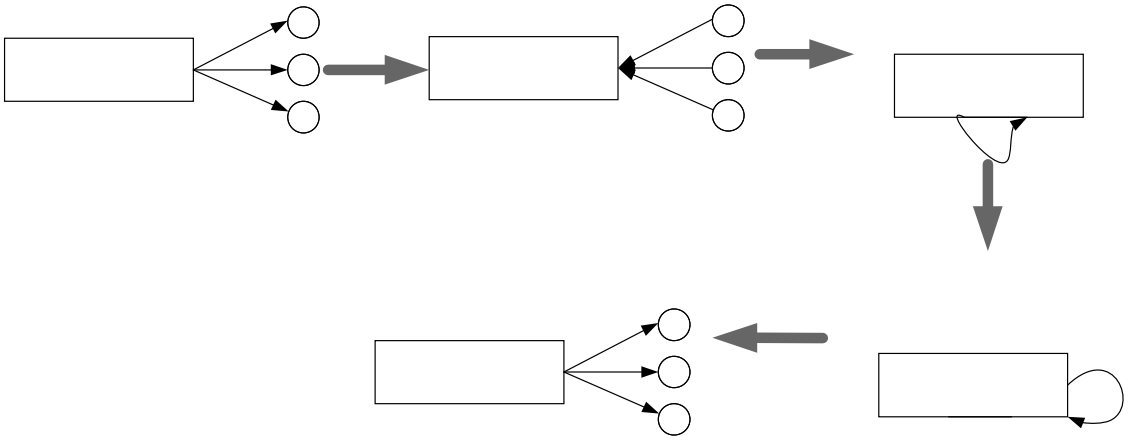


Figure 1. MLRT System Architecture

Each subsystem has two stages during training, the accuracy detection stage and the network creation stage.

3.1 Training

3.1.1 Accuracy Detection

The training data would be sent to the three subsystems. Each subsystem would return a percentage value, i.e. percentage accuracy (PA), indicating what fraction of the total dataset can be classified accurately using that subsystem, and the list of patterns that can be trained accurately, i.e. the indices.

For PNN and Backpropagation, the data is passed through the corresponding type of network to determine the PA and indices. The methodology used in BB for determining the PA and indices would be explained in the subsequent section.

3.1.2 Choosing a subsystem

In receiving these values, the main system would choose the subsystem to be used to create the network for the current recursion. It is desirable to have a PA close to 50% for training by BB and backpropagation, and a PA close to 100% for PNN. For BB, and backpropagation, 50% is chosen so a significant amount of data is trained in this recursion, with enough patterns left over for the next recursion. For PNN, since it is an approximation curve, having a large amount of data being classified using the curve shows that this is a dataset suitable for a curve fitting method, 100% being the ideal value. Therefore the minimum difference is taken – the difference between backpropagation and 50%, BB and 50% and PNN and 100%, indicating the closeness of the respective PAs to their ideal values, and the corresponding subsystem is chosen. However, when the BB returns an accuracy of less than 10% during the first recursion, then BB is chosen, because there is a chance that this is noise, and can be separately labeled and eliminated by BB. In the case that there is a close competition between high values (above 90%) between Backpropagation and PNN, backpropagation is chosen, since both approaches suit the dataset, and backpropagation is more widely used for classification problems. On the other hand, when dealing with a similar condition for regression problems, ability of PNN to fit a curve would favor its choice over backpropagation.

3.1.3 Sifting the data

Based on the indices returned by the chosen subsystem, data would be separated into the patterns used for the current recursion and the remaining data.

3.1.4 Network Creation

The former would be sent to the *network creation* block of the chosen subsystem. A network is created and stored.

The process is repeated with the remaining data. This continues until the data returned is too few. This is how data are trained.

3.2 Testing

During testing, usually one of a few networks is to be chosen from, for a test pattern. For each test pattern in the testing set, the network used by the training pattern closest to the test pattern [20] is determined. The test pattern is passed through this network to determine the output.

3.3 Bounding Boxes

The concept of BB is similar to clustering algorithms in unsupervised learning. However, in unsupervised learning, the data are not labeled but in BB, the data are. Furthermore, while many clustering algorithms try to find the most natural clusters, the objective of BB is to create linearly separable regions (or "boxes") containing only patterns from a single class. These boxes do not overlap with any other box, either for the same or a different class. Although the classes may not be linearly separable, the boxes are. If there are several regions of data of the same class that are separated by regions of other classes, BB would detect these boxes, and label the patterns separately. When passed through a

backpropagation network, since these regions are separable, classification would be made easier than that of the original dataset.

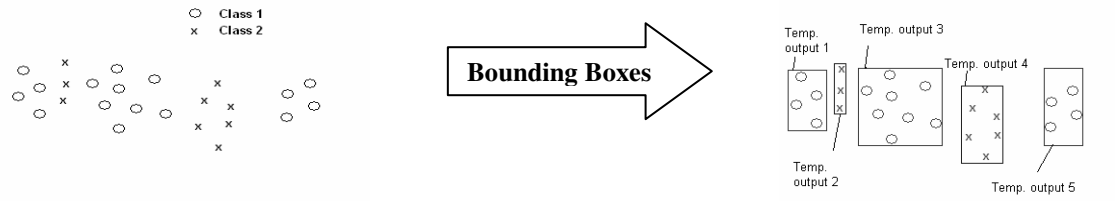


Figure 2. Modification by BB to make a problem dataset linearly separable

For the purpose of the following discussion, assume $\vec{X} = (x_1, x_2, ..., x_n)$ and $\vec{Z} = (z_1, z_2, ..., z_n)$ are n-dimensional vectors where n is the problem dimensionality.

$$\vec{X} < \vec{Z} \equiv \forall i \in 1...n, x_i < z_i \qquad (1)$$

The problem space can be visualized as an Euclidean space of n dimensions, where n is the number of inputs. A pattern is represented by $(\vec{X}, y)$ where $\vec{X}$ is the vector consisting of the inputs $(x_1, x_2, ..., x_n)$ and *y*, the scalar output. The vector $\vec{X}$ can be represented as a point in the Euclidean space. A box is represented by $(\vec{X}^{\min}, \vec{X}^{\max}, y, box\_label)$ where $\vec{X}^{\min}$ represents the minimum bounds of the box in each dimension and $\vec{X}^{\max}$ the maximum bounds. In the Euclidean space $\vec{X}^{\min}$ and $\vec{X}^{\max}$ represent the diagonals of a hyper-dimensional rectangle. The patterns are to be fit into boxes.

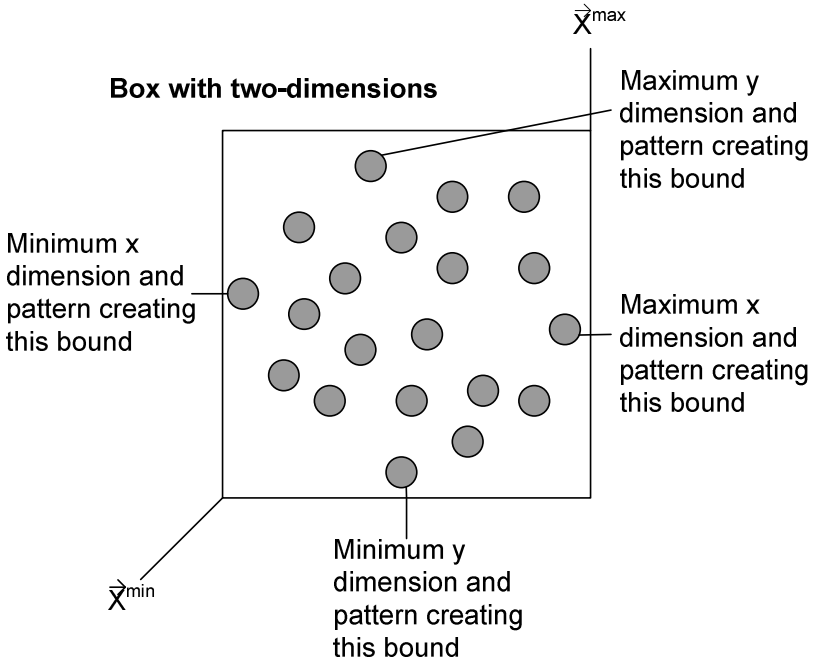


Figure 3. Fitting the patterns inside a bounding box

y is the class to which the patterns in the box (and therefore the box) belong, *box_label* is the unique label assigned to the box.

A pattern "is inside" a box if the point of the pattern fits within the boundaries of the box and can be represented as a function *f* mapping the patterns to the box.

Therefore:

$$f(\vec{X}1, y_1) = (\vec{X}2^{\min}, \vec{X}2^{\max}, y_2, box\_label2),$$

$$\text{iff } \vec{X}1 < \vec{X}2^{\max},\ \vec{X}1 > \vec{X}2^{\min},\ y_1 = y_2 \qquad (2)$$

Furthermore the boxes cannot overlap. Hence, for any two boxes $(\vec{X}1^{\min}, \vec{X}1^{\max}, y_1, box\_label1)$ and $(\vec{X}2^{\min}, \vec{X}2^{\max}, y_2, box\_label2)$, where *box_label*1 and *box_label*2 are labels of the box,

$$\overline{(\vec{X}1^{\min} < \vec{X}2^{\max}) \Lambda (\vec{X}2^{\min} < \vec{X}1^{\max})} \qquad (3)$$

The pseudocode for creating the boxes and putting the patterns inside the boxes while making sure the above conditions are satisfied is given below.

Table 1. Pseudocode for Bounding Boxes (BB)

```
1  for each pattern (X⃗, y1) in the problem set
2  if ∀boxes (X⃗^min, X⃗^max, y1, box_label), y1 ≠ y2
3  then new_box = create_new_box(pattern(i))
4  else
5   for each box (X⃗^min, X⃗^max, y2, box_label) | y1 = y2
6    if inside(box, pattern(i)) then f(X⃗, y1) = (X⃗^min, X⃗^max, y2, box_label),
7    else expanded_box = expand(box, pattern);
8      for every other box
9        if overlap(expanded_box, box)
10       then overlap = true; break;
11       end
12       if overlap = true
13       then new_box = create_new_box(pattern(i))
14       else store(expanded_box);
15       end
16     for each box (X⃗^min, X⃗^max, y2, box_label) | y1 ≠ y2
17     if inside(box,pattern(i)) then cut_up(box); end
18     end
```

The procedures mentioned above are explained below. The bold lettering indicates the line in the pseudocode where the procedure first appears.

1. **Line 3:** *new_box = create_new_box(pattern)*

   Assuming $pattern = (\vec{X}, y)$

   $new\_box = (\vec{X}^{\min}, \vec{X}^{\max}, y', box\_label) \mid \vec{X}^{\min} = \vec{X},\ \vec{X}^{\max} = \vec{X},\ y' = y,$

   box_label = no. of boxes + 1

2. **Line 6:** *inside(*box*, pattern(i))*

   if equation 2 is satisfied, return true, else return false.

3. **Line 7:** *expanded_box = expand*(box, pattern)

   Assume $pattern = (\vec{X}, y),\ \vec{X} = (x_1, x_2, ..., x_n),\ box = (\vec{X}^{\min}, \vec{X}^{\max}, y, box\_label)$

   Note that this procedure would be executed only if ~(*inside*(*box,pattern*))

   $\therefore (\exists i \in (1...n) \mid x_i < x_i^{\min}) \vee (\exists i \in (1...n) \mid x_i > x_i^{\max})$

   if for any $i \in 1..n,\ x_i < x_i^{\min}$, then $x_i^{\min} = x_i,$

if for any $i \in 1..n$, $x_i > x_i^{max}$, then $x_i^{max} = x_i$,
$expanded\_box = (\vec{X}^{min}, \vec{X}^{max}, y, box\_label)$

4. **Line 9:** *overlap (expanded_box,* box*)*

if equation 3 is satisfied then return true, else return false.

5. **Line 17:** *cut_up*(box)

The idea is to try and cut the boxes such that the sub-boxes are the largest they can be.

Let a pattern, $p: (\vec{X}inp, y_1), \vec{X}inp = (xinp_1, xinp_2, ..., xinp_n)$, which falls inside a box, $b: (\vec{X}^{min}, \vec{X}^{max}, y_2, box\_label) \mid \ y_1 \neq y_2$. Let each pattern in the box be $(\vec{X}, y_2)$, $\vec{X} = (x_1, x_2, ..., x_n)$.

for i:num_dimensions
  maxdiff(i) = $\max(\max(x_i^{max} - x'_i \mid x'_i > xinp_i), \max(x''_i - x_i^{min} \mid x''_i < xinp_i))$
end
find i | maxdiff(i)=max(maxdiff)
box b gets split into $b1: (\vec{X}1^{max}, \vec{X}1^{min}, y_2, box\_label1)$ and $b2: (\vec{X}2^{max}, \vec{X}2^{min}, y_2, box\_label2)$ such that
$\vec{X}1^{min} = (x_1^{min}, x_2^{min}, ..., x_i', ..., x_n^{min}), \vec{X}1^{max} = \vec{X}^{max}$;
$\vec{X}2^{min} = \vec{X}^{min}, \vec{X}2^{max} = (x_1^{max}, x_2^{max}, ..., x_i'', ..., x_n^{max})$

The above algorithm would be illustrated with a hypothetical 2-class dataset. The algorithm proceeds pattern by pattern. The shaded pattern in class 1 represents a pattern that has been placed inside a box.

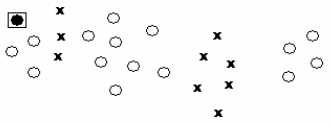
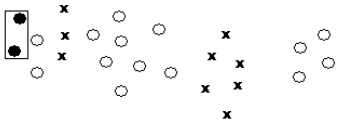
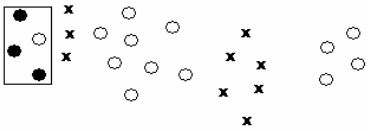
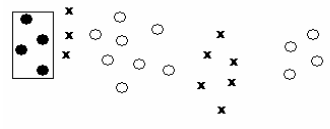
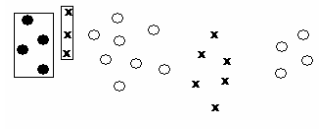
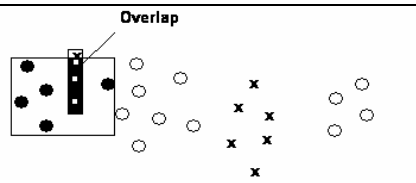
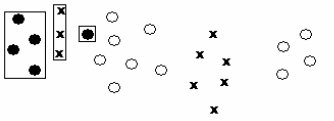
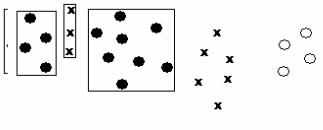
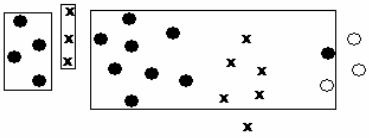
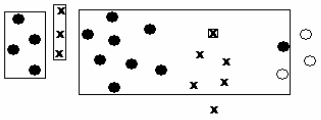
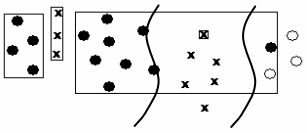
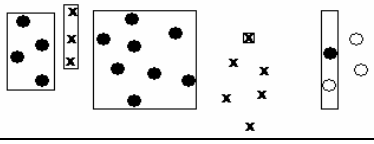
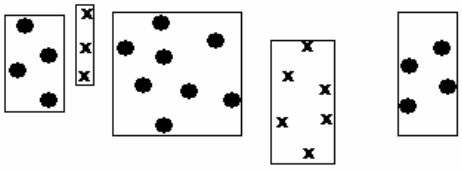
<table>
<tr><td>  </td><td>  </td><td>  </td></tr>
<tr><td>Step 1: Since the pattern has no boxes having the same class as itself, a box is created containing only that pattern. The maximum and minimum bounds of the box are equal to the input vector of the pattern. (lines 2-3)</td><td>Step 2: The box is expanded and checked to see if there are any overlapping boxes. Since there are none, the expanded box is stored. (lines 7 and 14)</td><td>Step 3: Once again, the box is expanded, and since there is no overlapping boxes, the expanded box is stored. (lines 7 and 14)</td></tr>
<tr><td>  </td><td>  </td><td>  </td></tr>
<tr><td>Step 4: Since the pattern already falls within the boundaries of the box, it is put inside the box. (lines 5-6)</td><td>Step 5: Similar to the steps described earlier, a second box of class 2 is created and expanded.</td><td>Step 6.1: When the box is expanded an overlap with another box is detected. Hence the box is not expanded. (lines 7-11)</td></tr>
<tr><td>  </td><td>  </td><td>  </td></tr>
<tr><td>Step 6.2: Instead a new box is created (lines 12-13)</td><td>Step 7: The box is expanded in steps similar to those shown earlier.</td><td>Step 8: The box is expanded to accommodate the incoming pattern, and since there are no overlapping boxes, the expanded box is stored. (lines 7 and 14)</td></tr>
<tr><td>  </td><td>  </td><td>  </td></tr>
<tr><td colspan="3">Step 9: If the existing box of class 2 is expanded, there would be an overlap therefore a new box of class 2 is created. (Above first two images. Similar scenario to step 6, therefore not re-explained here.) (lines 7-13) Since the new box falls into a box of an opposing class (third picture – box of class 2 falls into box of class 1), the larger box is cut up, such that there is no overlap. (lines 16-18)</td></tr>
<tr><td>  </td><td colspan="2">Using the steps as shown before, the boxes are expanded till the entire dataset is bound by boxes.</td></tr>
</table>

Figure 4. Explanation of Bounding Boxes (BB) using a hypothetical dataset. The italics denote the line in the pseudocode (Table 1) that is being executed.

Since the idea is to create a few regions such that the problem is more separable, the number of boxes should not be too large. If there are many boxes containing very few

patterns, there would be unnecessarily more classes, which need not be representative of the testing patterns. Hence only boxes that contain more than 5% of the entire dataset are chosen. The PA of BB is the percentage of data in these larger boxes. When this threshold was chosen larger the classification error was higher. When the threshold was smaller, BB was consistently chosen over the other subsystems, and still had higher classification error.

For the patterns in the box, the *box_label* is used instead of class (*y)*, as temporary outputs, and used in the backpropagation network, during the network creation stage. The linear separability due to the boxes makes it simpler to classify these patterns.

4. Theoretical Analysis

This section examines theoretically some aspects of the MLRT system. We give proof of convergence and examine the algorithm complexity.

4.1 Convergence

In this section, the convergence of MLRT is proved by showing that, given the decomposition algorithm presented in this paper, the partial derivatives of each of the subsystems converge to zero. Table 2 explains the notations used in this section

Table 2: Table of notations

In this section the following notations are used:
E– total error in all the patterns
E_{learnt} – error in the learnt patterns
$E_{unlearnt}$ – error in the unlearnt patterns
$\vec{W}$ - the weights vector

$$E = E_{learnt} + E_{unlearnt} \qquad (4)$$

However, the unlearnt patterns are not counted for the current recursion. The unlearnt patterns are hence taken out of the current recursion, in the gradient descent method. Therefore:

$$E = E_{learnt} + C \qquad (5)$$

where C is a constant. Therefore, due to the gradient descent method,

$$\frac{dE}{d\vec{W}} = \frac{dE_{learnt}}{d\vec{W}} \rightarrow 0 \qquad (6)$$

For the recursions where BB is chosen, the separability of the boxes, and therefore patterns in the boxes ensure E$\rightarrow$ 0 as $\frac{dE_{learnt}}{d\vec{W}} \rightarrow 0$, since backpropagation is used to classify the patterns in the network creation stage.

When backpropagation is chosen, since the structure of the network used in the network creation stage, and the structure of network used in the accuracy detection stage is the same, and the patterns are chosen, such that all the patterns are accurately classified, E$\rightarrow$0 as $\frac{dE}{d\vec{W}} \rightarrow 0$, since backpropagation is a gradient descent method.

In the case of the, the learnt patterns that are extracted for network creation are patterns that are accurately classified by the accuracy detection stage. As the same underlying network is being reused in the network creation stage, $E_{learnt} = 0$.

Since the networks created by each learner converge to zero error, the convergence of MLRT is ensured.

4.2 Complexity

Since the framework of MLRT is similar to that of RPHP [14] with the exception that there are three separate learners in MLRT as opposed to GA in RPHP, this section compares the complexity of MLRT with that of RPHP.

Table 3: Table of notations used in this section

In this section the following notations are used.
I – number of inputs
O – number of outputs
Complexity
C_{BB} - complexity of accuracy detection of BB subsystem
C_{BP} - complexity of accuracy detection of Backpropagation subsystem
C_{PNN} - complexity of accuracy detection of PNN subsystem
C_{NC} - complexity of the network creation stage
C_{GA} - complexity of GA training
C_{LM} - complexity of obtaining local minima
Bounding Boxes
B – number of boxes in BB
Backpropagation
E_{BP} - number of epochs of backpropagation, which is set to 50
T - the computational complexity for the sigmoid function $\frac{1}{1+e^{-x}}$
H_{BP} – the number of hidden nodes in backpropagation network– 6
Genetic Algorithms
E_{GA} - number of epochs for GA which is set to 20
N - population size which is 20
H_{GA} - number of hidden nodes in the network, varies from 10 to 50; assumed to be 35 on average
O - number of outputs

The complexity of MLRT can be expressed as:

$$C_{BB} + C_{BP} + C_{PNN} + C_{NC} \qquad (7)$$

We can express the complexity of RPHP by the following equation:

$$C_{GA} + C_{LM} \qquad (8)$$

Calculation of C_{BB}:

In the BB algorithm, for each pattern the following comparisons are made:

1. Check if the pattern belongs to a box - Compare the inputs of the pattern with the dimensions of each box of the same class. This occurs for all patterns.
2. Check for overlap between boxes in the case of expansion of boxes - Compare the inputs of the pattern with the dimensions of all the boxes. This happens if patterns don't fit into any box.
3. Check if pattern falls into a box of a different class – Compare the inputs of the patterns with the dimensions of each boxes of a different class. This occurs for patterns for which a separate box is created.

In addition there is additional complexity involved in cutting up the boxes in the case that a pattern does fall into a box of a different class. Since in most cases the box is divided into two separate boxes instead of several boxes, we can assume that this complexity is negligible.

Further, considering the worst case scenario, and therefore assuming that patterns undergo all the three comparisons, each pattern firstly undergoes one comparison with all boxes of the same class, secondly one comparison with all boxes of different classes, and thirdly one comparison with all boxes. The first two comparisons are complements, and hence, there are totally two comparisons with all boxes.

Comparison between a pattern and a box involves comparison between the minimum bounds of the box and the patterns, and comparison between the maximum bounds of the box and the patterns. Therefore one comparison with all boxes can be represented as:

$$2 \times I \times B$$

Since there are two comparisons with all boxes, the complexity is expressed by the following equation:

$$2 \times 2 \times I \times B \qquad (9)$$

Therefore for each pattern the complexity is 4*IB*.

Calculation of C_{BP} *:*

Assume that the forward propagation for the output is much higher than the backward propagation of error, in backpropagation,

The complexity of forward propagation is:

$$E_{BP} T(H + O) \qquad (10)$$

Calculation of C_{PNN} *:*

PNN follows the RBF structure, as PNN is an extension of the RBF for classification problems. In a RBF, a matrix is created such that

$$A = O(I^{'} I)^{-1} I' \qquad (11)$$

where $(I^{'} I)^{-1} I'$ is the pseudo inverse of *I*. (I' is the transpose of I).

Computational complexity of calculating pseudo inverse is I^3. Therefore, the complexity of PNN is:

$$C_{PNN} = OI^3 \qquad (12)$$

Calculation of C_{GA}:

The complexity of GA is:

$$TN(H+O)+T(E_{GA}-1)(H+O) \qquad (13)$$

The first term of this equation corresponds to the initial epoch, and the second term corresponds to the subsequent epochs.

Comparison with RPHP

Assuming that $C_{NC} \sim C_{LM}$,

the complexity of MLRT is less than that of RPHP when:

$$4IB \times P + OI^3 \times P + E_{BP}T(H+O) \times P < TN(H+O) \times P + (E_{GA}-1)T(H+O) \times P \qquad (14)$$

P is the number of patterns, which can be cancelled out on both sides. Therefore, for small values of I, the $E_{BP}T(H+O)$ term would be much higher than $4IB + OI^3$ terms due to the increased complexity of the T factor.

Therefore, neglecting the earlier terms:

$$E_{BP}T(H+O) < TN(H+O) + (E_{GA}-1)T(H+O)$$

$$Or: E_{BP}T(H+O) < T(H+O)\,[\,N + (E_{GA}-1)] \qquad (15)$$

It is clear that this equation would be satisfied if there are not a very large number of outputs. Therefore was datasets with small input dimensions (needed for equation 15 to hold), the complexity of MLRT is less than the complexity of RPHP. For datasets with

high input dimension, there is an exponential increase in the complexity of MLRT. Later in the paper, we verify this statement empirically.

5. Simulation Results

All datasets chosen for the simulation, except TWO-SPIRAL were obtained from the UCI Machine Learning Repository [21]. MLRT was tested with four datasets, namely SPAM, TWO-SPIRAL, SEGMENTATION and VOWEL. Ten trials were run and the training time and testing accuracy was noted in each case. The problem datasets were chosen such that they have a varying number of inputs and outputs, as well as training, testing and validation patterns available. Furthermore these datasets choose different combinations of subsystems for training.

5.1 Training Parameters

This section starts with the explanation of some parameters. After that a table of parameters is provided, followed by a comparison between the current parameters and other parameters used.

In BB, since the selection criteria for patterns determining PA are those in larger boxes, the threshold for selecting boxes is the fraction of patterns in the boxes above which the box is selected. The outputs for each network are either represented as a binary number where if the pattern is in class k, the kth digit of the binary number is 1, and other digits are all 0, or as decimals. For determining the output of the test pattern through the network, winner-takes-all algorithm is adopted where the index of the largest digit is

taken as the class which the pattern belongs to. The backpropagation network created in Network Creation has one hidden layer, in addition to the input and output layers. In PNN, a spread is the extent to which the curve is an approximation curve. A spread near 0 would result in the network acting as a nearest neighbor classifier. As the spread gets larger, the function would be an approximation curve rather than an interpolation curve.

Table 4. Training Parameters

Subsystem Used	**Parameters**
Bounding Boxes (BB)	Accuracy detection – Threshold for selection of boxes – 0.05 Output format – binary Method of determining output - Winner takes all Network Creation – Hidden nodes – 6 Number of epochs – 50 Output format – binary Method of determining output - Winner takes all
Backpropagation	Accuracy detection – Hidden nodes – 5 Number of epochs – 50 Output format – binary Method of determining output - Winner takes all Network creation – Hidden nodes – 5 Number of epochs – 50 Output format – binary Method of determining output - Winner takes all
Probabilistic Neural Networks (PNN)	Accuracy detection – Spread – 0.9 Output format – decimal Network Creation – Spread – 0.95 Output format – decimal

Substituting this set of values, we can rewrite equation 15 as

$$250T + 50OT < 1365T + 39OT \qquad (16)$$

yielding a maximum value of O as 101.36.

It was noted that these values yield the optimal results. Several other values were compared. When the spread of the PNN reduces, the curve fitting becomes an interpolation, rather than an approximation curve fitting. While choosing the most

appropriate subsystem, if a large amount of data can be classified correctly with an approximate curve, then the most natural method to use in classification would be curve fitting; hence PNN is chosen over other methods. However, when the spread is reduced, for any problem set, a large amount of data is classified correctly with the PNN, as the interpolation results in fitting the curve to suit the dataset. Testing accuracy however, falls, as an interpolation curve results in overfitting.

Output format was chosen to be binary for backpropagation. This is ideal for a classification problem, as linear separability plays an important role especially in BB, in classifying the patterns. In case of decimal outputs, regression rather than classification is adopted.

The results and comparison with other benchmark algorithms are given as follows:

Multisieving [23], constructive backpropagation (CBP) and RPHP are compared with MLRT. Comparison with CBP illustrates the importance of the multi recursive approach as opposed to the single stages training approach. Multisieving is an algorithm that implements recursive learning using only neural networks (single learner). RPHP uses a hybrid combination of two learners, while MLRT combines three learners to overcome the computational intensity of the GA based learner. Two versions of RPHP are presented. RPHP-MGGD uses minimal coded genetic algorithms to reduce the training time, while RPHP-GAD reflects the training time using standard GAs.

A. Spam

Table 5. Comparison of MLRT results in SPAM dataset with other benchmark algorithms

Algorithm used	Training time (s)	Classification error (%)
Constructive Backpropagation	43.649	27.92
Multisieving with KNN Pattern distributor	123.12	21.06
RPHP – GAD	156.81	20.75
RPHP – MGGD	82.803	20.97
MLRT	***45775***	***7.204***

B. Two-Spiral

Table 6. Comparison of MLRT results in TWO-SPIRAL dataset with other benchmark algorithms

Algorithm used	Training time (s)	Classification error (%)
Constructive Backpropagation	15.58	49.38
Multisieving with KNN Pattern distributor	35.89	23.61
Dynamic Topology Based Subset Selection (TSS)	-	28.0
RPHP– GAD	87.91	10.54
RPHP– MGGD	59.97	11.08
MLRT	***15.745***	***12.37***

C. Segmentation

Table 7. Comparison of MLRT results in SEGMENTATION dataset with other benchmark algorithms

Algorithm used	Training time(s)	Classification error (%)
Constructive Backpropagation	693.8	5.74
Multisieving with KNN Pattern distributor	760.64	7.28
Output Parallelism	2219.2	5.44
RPHP– GAD	2219.2	5.44
RPHP– MGGD	1151.8	4.32
MLRT	***610.87***	***6.544***

D. Vowel

Table 8. Comparison of MLRT results in VOWEL dataset with other benchmark algorithms

Algorithm used	Training Time (s)	Classification error (%)
Constructive Backpropagation	237.9	37.16
Multisieving with KNN Pattern distributor	318.23	39.43
Output Parallelism	418.9	25.54
Output Parallelism with Pattern Distributor	534.3	24.89
RPHP– GAD	842.16	16.72
RPHP– MGGD	473.88	17.73
MLRT	***189.586***	***21.369***

It is noted that, with the exception of the SPAM dataset, MLRT has a generally shorter training time than the other algorithms with comparable generalization accuracy. The large training time of MLRT on the SPAM dataset can be attributed to the BB component of the algorithm, the complexity of which increases with input dimension. To reduce this exponential effect on training time, we substitute the BB component of MLRT with clustering.

6. Comparison with clustering

Since BB is a revised form of clustering conventionally used in unsupervised training, in the system, BB was replaced in the following by conventional clustering as an effort to reduce training time, and the results were compared. AHC, one of the common means of clustering was incorporated, with single linkage. This section provides the comparison and analysis of the results.

Generally the system with BB is more accurate than the one with clustering, albeit the tradeoff in training time. In creating the natural clusters, there is no regard given to linear separability, unlike in BB. As a result, although there is a separation of the problem set, there is no simplification, and therefore, the results are not as good as BB.

Table 9. Comparison of results between the MLRT system with BB and when BB is replaced by clustering

Dataset	BB error (%)	Clustering error (%)	Training time BB (s)	Training time with Clustering (s)
Spam	7.204	9.85	45775	534.562
Two-Spiral	12.37	12.37	15.745	10.1704
Vowel	21.37	20.32	189.59	328.407
Segmentation	6.544	23.47	610.87	433.083

However, it was also noted that while BB has a large training time overhead associated with the increase in accuracy in spam, such an overhead is not present with clustering.

This is because in order to ensure separability, BB separates the datasets in each dimension. Therefore, with an increase in the number of inputs, there is a large increase in time. Clustering on the other hand works based on the distance between the patterns in the dataset, and is not dimension dependent. Hence clustering can replace BB in cases where the number of inputs is large, to circumvent the time overhead.

7. Heuristics to improve the MLRT algorithm

Two variants of the basic MLRT system, MLRT-2 and MLRT-3 would be described here. MLRT-2 explores a different subsystem selection algorithm while MLRT-3 proposes an improvement to the Bounding Boxes algorithm.

MLRT-2

In the second version of the system, subsystem selection for a recursion of MLRT was changed. Unlike the basic MLRT system, there are no cut-offs, but the amount of increase or decrease in points depends on the degree to which the PA returned from the subsystems indicating the suitability of the subsystem for the dataset or subset. All subsystems are assigned 50 points and depending on a set of rules, we reward or punish the subsystem and therefore change its points. The subsystem with the highest number of points wins. The subsystems can have a maximum of 100 points and a minimum of 0 points. The increase and decrease is relative, and this value was chosen due to mathematical simplicity, since the PAs also range from 0 to 100.

MLRT-3

In this version, change is made to the network creation stage in BB. The network represents the correlation between the validation data and the existing boxes. The parameters chosen to represent the correlation between the patterns and the boxes are such that they represent the variance of patterns within the boxes, significance of the boxes, and proximity of the boxes with the patterns. To this end, the three parameters chosen are (1) distance from the validation pattern to the centroid of the box, (2) mean distance between (training) patterns in the box and (3) number of (training) patterns in the box.

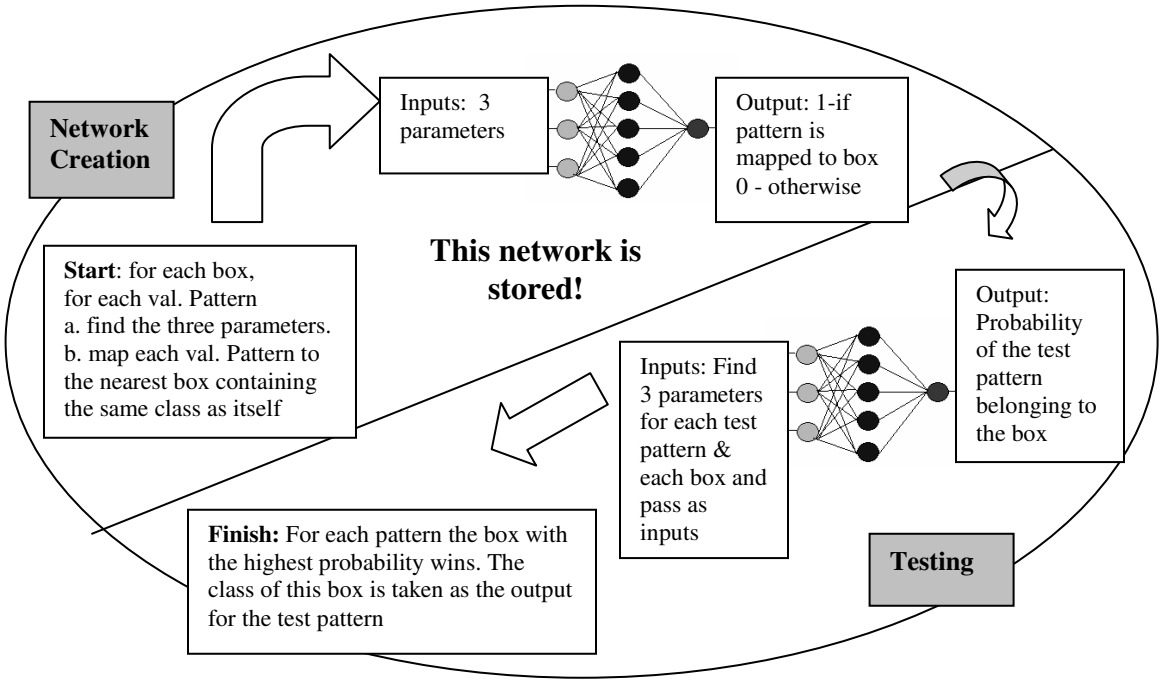


Figure 5. Changes made in Network Creation and Testing stages of Version 3

A table comparing the results for the two versions is given below.

Table 10. Results of MLRT, MLRT-2 and MLRT-3

Dataset	MLRT		MLRT-2		MLRT-3	
	C. error (%)	Tr. time(s)	C. error (%)	Tr. Time (s)	C. error(%)	Tr. Time (s)
Spam	7.204	45775	9.63	31791	9.51	31772
Two-Spiral	12.37	15.745	12.37	21.50	12.37	21.32
Vowel	21.37	189.59	20.41	230.03	24.98	211.47
Segmentation	6.544	610.87	5.904	615.31	6.24	610.78

Although there is not much change in the results, version 2 was deemed significant because of scalability. When there is an increase in the number of subsystems, the rules can be added, and the points modified, without affecting the existing rules. In version 3, since there is only a smaller and simpler network, there may be a decrease in the training time. Further, there is a closer correlation between the training and testing accuracy. Therefore, if the training accuracy is improved, there is a chance that the testing accuracy is also improved**.**

These systems were introduced based on some heuristics and further work is required before they can be perfected. In MLRT 2, for instance, more rules are required to cater completely to the three subsystems. Similarly, in MLRT 3, other variables than other used currently may be able to give a better representation of the dataset. It is expected that these additions will improve the performance of the MLRT versions 2 and 3.

8. Discussions

The strategy used by MLRT is to build several subsystems, and cater to the underlying characteristics of these three subsystems. Three basic subsystems were chosen based on basic features in general datasets to explore the viability of MLRT. More complex versions of the systems were deemed unnecessary at the initial stage of research. Besides, using complex versions might lead to an increase in time overhead. Unlike a single-learner approach, when one learner is not able to classify one problem correctly, MLRT can afford to rely on other systems that would possibly perform better.

Generally, MLRT is able to identify the underlying characteristics of the datasets and use them in classifying the data. In SPAM, it is able to deduce that classification would be done more accurately than the previous algorithms if a small portion of the dataset was removed before training the rest of the data, and this leads to an increase in accuracy over all previous methods. The training of TWO-SPIRAL works on the fact that there is an underlying function within the data, and MLRT uses that function, hence training the data with only one recursion as compared to two recursions on RPHP.

For the majority of the datasets tested, the performance is on par with RPHP, while training time is greatly reduced and comparable to that of Constructive Backpropagation, as features of the datasets have been used in training the patterns in a simple and more efficient manner. However, despite the significant increase in accuracy in SPAM, there is a very large time overhead, owing to the large number of inputs. It is noticed that when BB is replaced by clustering, there is a steep drop in this overhead, while the accuracy is still comparable. Since this is the first version of BB, it is not yet tuned to perform quickly on datasets with a high dimensionality. It has been shown however, that with simple replacement and improvement, this overhead can be greatly reduced.

This is the initial stage of MLRT research. Although MLRT accuracy rate is not substantially higher than RPHP at this juncture, the current version has nevertheless shown that performance can be on par, in most cases with RPHP. Furthermore, MLRT has demonstrated on SPAM, that it has ‘understood’ the nature of the dataset, and uses it to accurately classify results, having results superior to other datasets. In SPIRAL it is

able to show that dataset-specific training by choosing subsets accurately can greatly reduce efficiency.

Since the genetic algorithm component of RPHP works as a black box search, modifications to improve the accuracy of the algorithm can only be made outside the black box. However, with the case of MLRT, research can be performed at the subsystem level to improve the flexibility and the adaptability of the algorithm.

The current subsystems on MLRT allow the system to decipher the dataset better, in addition to training the network using the dataset. As a result, using these basic functions, one is able to find out more about the dataset, hence enabling the building of more complex learners from these simple barebones, that may be able to cater to more complex datasets.

9. Conclusion and future work

This paper has demonstrated that using a multi-learner based approach to neural networks that chooses the learners based on the problem dataset produced increased training efficiency without compromising on accuracy. Although the performance differs for different problems, since the subsystems provide information about the datasets, and the methodology used for learning is transparent, there is scope for expansion and future improvement. Additional learners can be added, based on the information gathered about the datasets.

Multi-learner based recursive approach attempts to cater to as many datasets as possible. Hence, with more research done on different types of datasets, and using the system to find out more about datasets, more can be learnt to build efficient subsystems to this end. MLRT can further be enhanced for regression, and function optimization problems. In addition, to cater to a particular set of data whose characteristics are more or less known to the user, a set of custom made algorithms can be fit into the basic MLRT system.

References

[1] Rumelhart DE, Hinton GE and Williams, RJ, "Learning internal representations by error propagation". *Parallel Distributed Processing*, vol. 1, MIT Press 1986, pp. 318-362

[2] Ash T, "Dynamic node creation in backpropagation networks", *ICS Report 8901*, UCSD Feb. 1989

[3] Montana DJ, "A Weighted Probabilistic Neural Network", *Advances in Neural Information Processing Systems 4*, 1992, pp. 1110-1117

[4] Specht DF, "Probabilistic Neural Networks", *Neural Networks,* Vol. 3, Issue 1, 1990, pp. 109-118

[5] Wassermann PD, "Advanced Methods in Neural Networks", *Van Nostrand Reinhold*, New York, 1993, pp. 35-55

[6] Jain AK, Murthy MN and Flynn PJ, "Data Clustering: A Review", *ACM Computing Reviews*, 1999

[7] Fred A and Jain AK, "Combining Multiple Clustering Using Evidence Accumulation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 6, 2005, pp. 835-850

[8] Topchy A, Minaeim B, Jain AK and Punch W, "Adaptive clustering Ensembles", *Proceedings of the International Conference on Pattern Recognition*, vol. 1, Cambridge, 2004, pp. 272-275

[9] Steinbach M, Karypis G and Kumar V, "A Comparison of Document Clustering Techniques", *KDD Workshop on Text Mining*, 2000, pp. 1-2

[10] Kanugo T, Mount DM, Netanhayu NS, Piatko C, Silverman R and Wu AY, "The Analysis of a simple K-Means Clustering Algorithm", *Symposium on Computational Geometry*, 2000, pp. 100-109

[11] Wu KL and Yang MS, "Alternative c-means Clustering Algorithms", *Pattern Recognition,* vol. 35, no. 10, 2002, pp. 2267-2278

[12] Vesanto J and Alhoniemi E, "Clustering of the Self-Organizing Map", *IEEE Transactions on Neural Networks*, 2000, pp. 586-600

[13] Flexer A, "On the use of self-organizing maps for clustering and visualization" *Intelligent Data Analysis,* vol. 5, no. 5, 2001, pp. 373 – 384

[14] Guan SU and Ramanathan K (2004), “Recursive Percentage based Hybrid Pattern Training for Curve Fitting”, *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems*, pp. 445-450
[15] Chen S, Cowan CFN and Grant PM, "Orthogonal Least Squares Learning Algorithm for Radial Basis Function Networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, 1991, pp. 302-309
[16] Park J and Sandberg IW, “Universal approximation using radial-basis-function networks”, *Neural Network Computation,* vol. 3, issue 2, 1991, pp. 246-257
[17] Guan SU, Neo TN and Bao C, “Task Decomposition using Pattern distributor”, *Journal of Intelligent Systems*, 2003, vol. 13, pp. 123-150
[18] Schaffer JD, Caruna RA and Eshelman LJ, “Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art,” *Proceedings of the IEEE Workshop on Combinations of Genetic Algorithms and Neural Networks*, 1992, pp. 1-37
[19] Montana DJ and L Davis, “Training Feedforward Neural Networks using Genetic Algorithms,” *Proceedings of the International Joint Conference on Artificial Neural Networks*, 1989, pp. 762-767
[20] Wong MA and Lane T, “A kth nearest neighbour clustering procedure”. *Journal of the Royal Statistical Soci*ety (B), 45(3), 1983, pp. 362-368
[21] Newman DJ, Hettich S, Blake CL, and Merz CJ, (1998) “UCI Repository of machine learning databases” *[http://www.ics.uci.edu/~mlearn/MLRepository.html]* Irvine, CA: University of California, Department of Information and Computer Science
[22] Guan S U, Li S (2002), “Parallel Growing and Training of Neural Networks Using Output Parallelism”, IEEE Trans. on Neural Networks, 13(3), pp542 -550.
[23] Lu B L, Ito K, Kita H, Nishikawa Y(1995), Parallel and modular multi-sieving neural network arcitecture for constructive learning, In proceedings of the 4th International Conference on Artificial Neural Networks. 409, 92-97.
[24] Meir R and Rätsch G (2003), An introduction to boosting and leveraging. Advanced lectures on machine learning, Springer, pp 119-184
[25] Schapire R E (1997), Using output codes to boost multiclass learning problems, Fourteenth international conference on machine learning, San Francisco, pp313-321
[26] Breiman L (1996), Bagging predictors, Machine learning 24(2), pp123-140.